This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A software architecture for debugging a distributed database application process on a client connection, comprising:

a server, wherein the server runs the distributed database application on a computer;

a client, wherein the client interacts with the distributed database application by way of the client connection;

a debugger, wherein the debugger debugs the distributed database application process, wherein the server, client, and debugger each run on separate computers; and

an application program interface (API), wherein the API receives a debugger request from the debugger to debug managed code, causes the server to call a debugging component, and wherein the debugger debugs the managed code.

2.-4. (Cancelled)

5. (Currently Amended) The software architecture of claim 1~~-4~~, wherein the API receives a debugger request to debug managed code, causes the server to call a remote debugging component, and wherein the debugger debugs the managed code by way of the remote debugging component.

6. (Original) The software architecture of claim 1, wherein the API further detects a transition between Transact-Structured Query Language (T-SQL) and managed code and calls a method to communicate the transition to the debugger.

7. (Original) The software architecture of claim 6, wherein the debugger debugs the distributed application process according to the transition.

8. (Original) The software architecture of claim 1, wherein the debugger further comprises a user interface, wherein the user interface displays only the T-SQL activity within the server on the client connection being debugged.

9. (Original) The software architecture of claim 1, wherein the debugger further comprises a user interface, wherein said user interface displays only threads associated with the client connection.

10. (Original) The software architecture of claim 1, wherein the server detects an addition of a dynamic T-SQL frame to a user stack within the server and calls a method to pass text of the dynamic T-SQL frame to the debugger.

11. (Currently Amended) A method of communicating between a server process, a client process and a debugger process in a distributed database environment, wherein the server process, the client process, and the debugger process each operate on separate computer, the method comprising:

receiving a first call for a stored procedure from the debugger process to debug managed code;

returning an interface pointer to the debugger process responsive to the received first call;

receiving a second call for a register method from the debugger process, wherein the second call comprises a machine name, a process ID and an interface pointer;

recognizing a client connection matching the machine name, process ID and interface pointer on the server process;

halting execution of the client connection on the server process responsive to said detection;

executing a third call, wherein the third call establishes operative communications between the debugger process and the client process; and

debugging the client process.

12. (Original) The method of claim 11, wherein the third call uses the interface pointer received in the second call.

13. (Original) The method of claim 11, further comprising:

detecting a request from the debugger process to debug managed code;

calling a remote debugging component; and

debugging the managed code by way of the remote debugging component.

14. (Original) The method of claim 11, further comprising detecting a transition between Transact-Structured Query Language (T-SQL) and managed code on the client connection; and calling a method to communicate the transition to the debugger process.

15. (Original) The method of claim 14, further comprising debugging the client connection according to the transition.

16. (Original) The method of claim 11, wherein the server process is executing T-SQL code on the client connection and the debugger process is debugging the T-SQL code, and further comprising displaying, on a user interface, only the T-SQL code executed by the server process on the client connection being debugged.

17. (Original) The method of claim 11, wherein the server process is executing managed code on the client connection and the debugger process is debugging the managed code, and further comprising displaying, on a user interface, only threads associated with the managed code being debugged.

18. (Original) The method of claim 11, further comprising detecting an addition of a dynamic T-SQL frame to a user stack within the server process and calling a method to pass text of the dynamic T-SQL frame to the debugger process, and wherein debugging the client process is by way of the text of the dynamic T-SQL frame.

19. (Currently Amended) A computer-readable medium having computer-executable instructions for performing a method of communicating between a server process, a client process and a debugger process in a distributed database environment, wherein the server process, the client process, and the debugger process each operate on separate computer, the method comprising:

receiving a first call from a computer for a stored procedure from the debugger process to debug managed code on a computer;

returning an interface pointer to the debugger process responsive to the received first call;

receiving a second call from a computer for a register method from the debugger process, wherein the second call comprises a machine name, a process ID and an interface pointer;

recognizing a client connection matching the machine name, process ID and interface pointer on the server process;

halting execution of the client connection on the server process responsive to said detection;

executing a third call, wherein the third call establishes operative communications between the debugger process and the client process; and

debugging the client process.

20. (Original) The computer-readable medium of claim 19, wherein the third call uses the interface pointer received in the second call.

21. (Original) The computer-readable medium of claim 19, wherein the method further comprises:

detecting a request from the debugger process to debug managed code;

calling a remote debugging component; and

debugging the managed code by way of the remote debugging component.

22. (Original) The computer-readable medium of claim 22, wherein the method further comprises detecting a transition between Transact-Structured Query Language (T-SQL) and managed code on the client connection; and calling a method to communicate the transition to the debugger process.

23. (Original) The computer-readable medium of claim 22, wherein the method further comprises debugging the client connection according to the transition.

24. (Original) The computer-readable medium of claim 19, wherein the server process is executing T-SQL code on the client connection and the debugger process is debugging the T-SQL code, and wherein the method further comprises displaying, on a user interface, only the T-SQL code executed by the server process on the client connection being debugged.

25. (Original) The computer-readable medium of claim 19, wherein the server process is executing managed code on the client connection and the debugger process is debugging the managed code, and wherein the method further comprises displaying, on a user interface, only threads associated with the managed code being debugged.

26. (Original) The computer-readable medium of claim 19, wherein the method further comprises detecting an addition of a dynamic T-SQL frame to a user stack within the server process and calling a method to pass text of the dynamic T-SQL frame to the debugger process, and wherein debugging the client process is by way of the text of the dynamic T-SQL frame.

27. (Currently Amended) A method of initiating a debugging session between a debugger and a client connection on a server running a distributed database application, comprising:

specifying the client connection of managed code to be debugged by way of an API, wherein the debugger, the client, and the server each are on a separate computer;

returning an interface pointer to the debugger by way of the API;

calling a register method, wherein the register method uses the interface pointer to detect the client connection associated with the interface pointer;

halting execution of the client connection; and

enabling the debugger to debug the client connection by way of the server and the API.

28. (Original) The method of claim 27, further comprising debugging the client connection.

29. (Original) The method of claim 27, further comprising returning a machine name and process identifier to the debugger by way of the API.

30. (Original) The method of claim 27, further comprising detecting a security context of the client connection and performing said connecting step only if the security context matches a predetermined security context.

31. (Original) The method of claim 27, wherein said calling step is by way of a distributed component object model (DCOM).

32. (Currently Amended) A computer-readable medium having computer-executable instructions for performing a method of initiating a debugging session between a debugger and a client connection on a server running a distributed database application, the method comprising:

specifying the client connection having managed code to be debugged by way of an API loaded on a computer, wherein the debugger, the client, and the server each are on a separate computer;

returning an interface pointer to the debugger by way of the API on the computer;

calling a register method, wherein the register method uses the interface pointer to detect the client connection associated with the interface pointer;

halting execution of the client connection; and

enabling the debugger to debug the client connection by way of the server and the API.

33. (Original) The computer-readable medium of claim 32, wherein the method further comprises debugging the client connection.